

Indexes

An index is an object that is stored in the database. An index is associated with one or more columns, in a specific table, that act as the key for the index. The DBMS uses the information stored in an index to speed up the lookup/retrieval of data stored in the table. Functionally an index object is similar to an index at the back of a book.

PostgreSQL automatically creates an index for each unique constraint and primary key constraint to enforce uniqueness. Thus, it is not necessary to create an index explicitly for primary key columns. Each entry in the index contains information (pointers) to where the associated row in the table is located. This information is then used by the DBMS to speed up retrieval of data from the table.

A non-primary key index provides a method of using a second criterion for quickly finding/retrieving rows from a table. For example, we could define a non-unique index for the Registration table that uses StaffID as the key of the index. The DBMS could use this index to quickly retrieve the rows for the staff involved in a specific registration.

You can, theoretically, have an unlimited number of indexes per table only constrained by practical constraints like storage and performance. There is a maximum of 32 columns per index.

Indexes speed up data retrieval, but they slow down the operations that modify data in a table. Consider a table that has a primary key index and three non- primary key indexes. To insert a new row in the table the DBMS must add the row to the table and update each of the indexes.

The DBMS implements the update operation by deleting the old row and adding a new row with the new data. To update an existing row in a table the DBMS deletes the target row from the table and then updates each of the indexes. The DBMS then adds the new row to the target table and again updates each of the indexes. Updating the index may involve

adding/deleting several records in the index, depending on the size of the index (based on the number of rows in the base table). This demonstrates the overhead involved in maintaining indexes.

Some database designers create a non-unique index for each foreign key. This can increase the performance of queries that retrieve data from two or more tables. This needs to be weighed against the increased overhead for modifying data in the table.

The Create Index statement creates an index object.

Create Index syntax (simplified)

Create [Unique] Index [If Not Exists] ***index_name***

On ***table_name*** (***column_name*** [, ...n] [Asc | Desc]);

Column name represents the key for the index. One or more columns may act as the key for the index. All indexes must have a unique name. In SDEV1201 we normally use the prefix IX when naming an index.

Example: create a non-unique index associated with the Registration table using StaffID (a foreign key referencing the Staff table) as the index key:

Create Index IX_Registration_StaffID

On Registration (StaffID);

Example: create a unique index associated with the Club table using ClubName (to ensure no club names are re-used) as the index key:

Create Unique Index IX_Club_ClubName

On Club (ClubName);

The Drop Index statement deleted an index object. Note: you cannot drop an index used for a Primary Key.

Drop Index syntax (simplified)

Drop Index [If Exists] ***index_name***;

Example: drop the Registration StaffID index just created

```
Drop Index IX_Registration_StaffID;
```